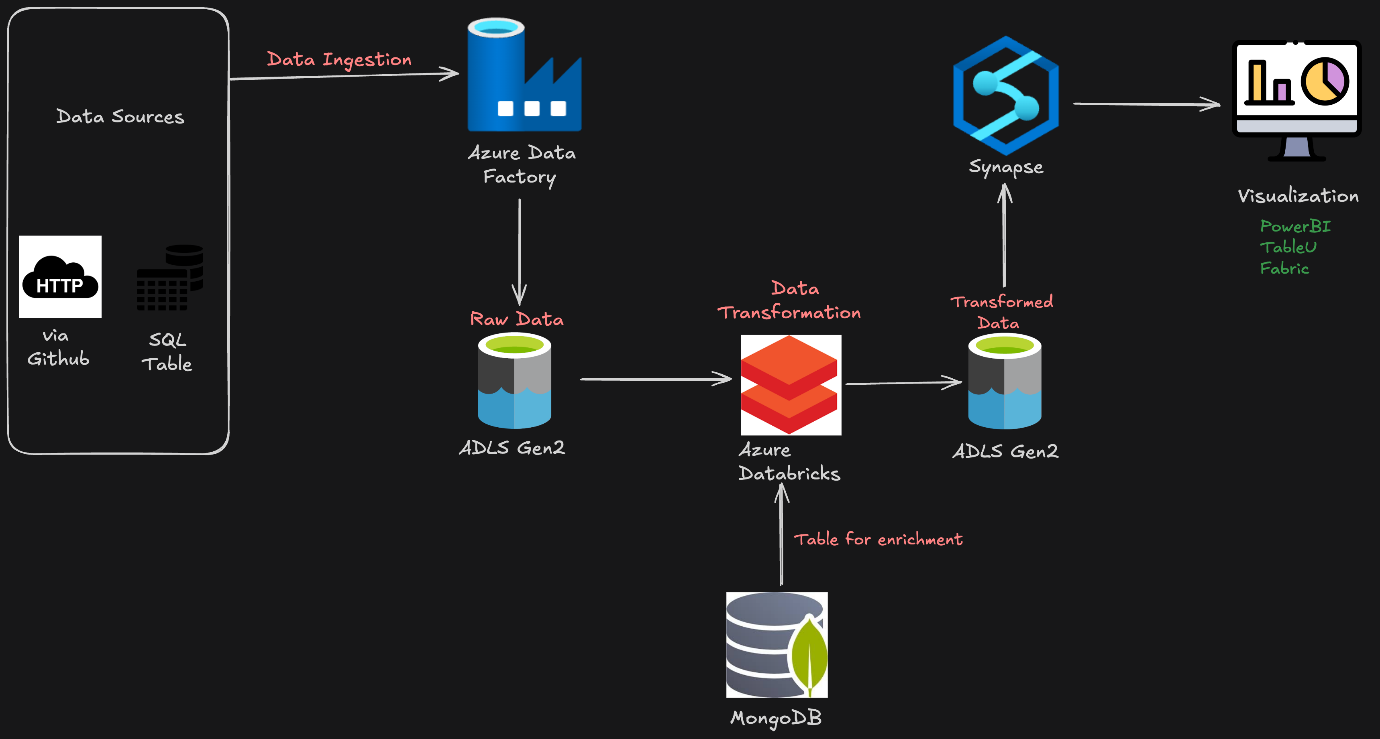
**Ecomm Project**

## Project Summary:

This project involves building a full-stack data engineering pipeline using Azure and a real-world e-commerce dataset (Olist). The pipeline implements the Medallion Architecture—Bronze (raw), Silver (cleaned), and Gold (curated)—using tools like Azure Data Factory (ADF), Azure Data Lake Storage (ADLS Gen2), Azure Databricks, MongoDB, Azure Synapse Analytics, and Power BI. It emphasizes hands-on cloud-based learning, real-world complexity, and modular pipeline design.

## Architecture:



1. **Raw Data Sources**
   1. Download the **Olist Brazilian E-commerce dataset** from Kaggle (<https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce/data> )
   2. Upload CSVs to:
   * **GitHub** (for HTTP ingestion - <https://github.com/mayank953/BigDataProjects/tree/main/Project-Brazillian%20Ecommerce/Data> )
   * **MySQL (**for SQL enrichment)
     + **Using files.io (**[**https://dash.filess.io/**](https://dash.filess.io/) **)**
     + **Creating olist**\_payment\_dataset using list\_order\_payments\_dataset.csv by using colab (<https://colab.research.google.com/drive/1bIfmyeGgdIzkeMl36W4zcuqzO3MkxjcD#scrollTo=JRuxo21hzuLb> )
   * MongoDB (for NOSQL)
     + **Using files.io (**[**https://dash.filess.io/**](https://dash.filess.io/) **)**
     + Creating product\_categories table using product\_category\_name\_translation.csv
2. Creating Azure Data Lake Storage
   1. Create storage account in any region
      1. Name should be universally unique
      2. Need to enable hierarchical namespace
   2. Create a container (Data + Storage)
      1. Container – ecomm
      2. Create directories
         1. Bronze
         2. Silver
         3. Gold
3. Creating Azure Data Factory
   1. Create data factory
   2. Click on “Lauch Studio”
      1. Go to author 🡪 click on create pipeline
         1. Add “for each” activity, to loop through all the files
         2. You can either
            1. Pass all the file names as an array to parameter - ["olist\_customers\_dataset.csv","olist\_geolocation\_dataset.csv","olist\_order\_items\_dataset.csv","olist\_order\_reviews\_dataset.csv","olist\_orders\_dataset.csv","olist\_products\_dataset.csv","olist\_sellers\_dataset.csv","product\_category\_name\_translation.csv"]
            2. Create lookup activity

Place json file with all file names in storage acc or HTTP

Pass array output of lookup **(@activity('ecomm\_lookup').output.value**) as input to for each loop

Create copy activity inside for each

Source:

Create LS for HTTP Source, provide only base URL (<https://raw.githubusercontent.com/>)

On the DS. Leave Relative URL empty

Click on open on the DS

Create parameter (input\_file)

Relative URL should be parameterized ([IVReddy81/Venkat\_Projects/refs/heads/main/ECOM\_End\_to\_End\_Project/csv\_files/**@{dataset().input\_file}**](mailto:IVReddy81/Venkat_Projects/refs/heads/main/ECOM_End_to_End_Project/csv_files/@%7bdataset().input_file%7d))

Now pass the “input file” parameter **(@item().file**) [file is key in json)

Sink:

Create LS for ADLS Sink, provide only directory but not file

Click on open on DS:

Create parameter (output\_fle)

Pass output\_file as parameter for file name (**@dataset().output\_file\_name)**

Now pass the “output\_file” parameter **(@item().file**) [file is key in json]

* + - 1. Click on Validate to check for Any errors
      2. Click on Debug to start the pipeline
      3. Click on Publish to save all the data

1. Create databricks service
   1. Click on “Launch WorkSpace”
   2. To connect ADLS and Databricks follow below process
      1. Go to “App Registartions” 🡪 Regiser New App
         1. Copy client id
         2. Copy Tenet id
      2. Go to “Manage”
         1. Click on “certificates & secrets” 🡪 create new secret
            1. Copy secret value
   3. Create a new notebook in databricks, use below code

# Define the variables for the ADLS connection

adls\_account\_name = "stoage\_account\_name"

adls\_container\_name = "ontainer\_name"

client\_id = "clinet\_id"

tenant\_id = "tenant\_id"

client\_secret = "secret\_value"

# Set the Spark configuration for ADLS connection

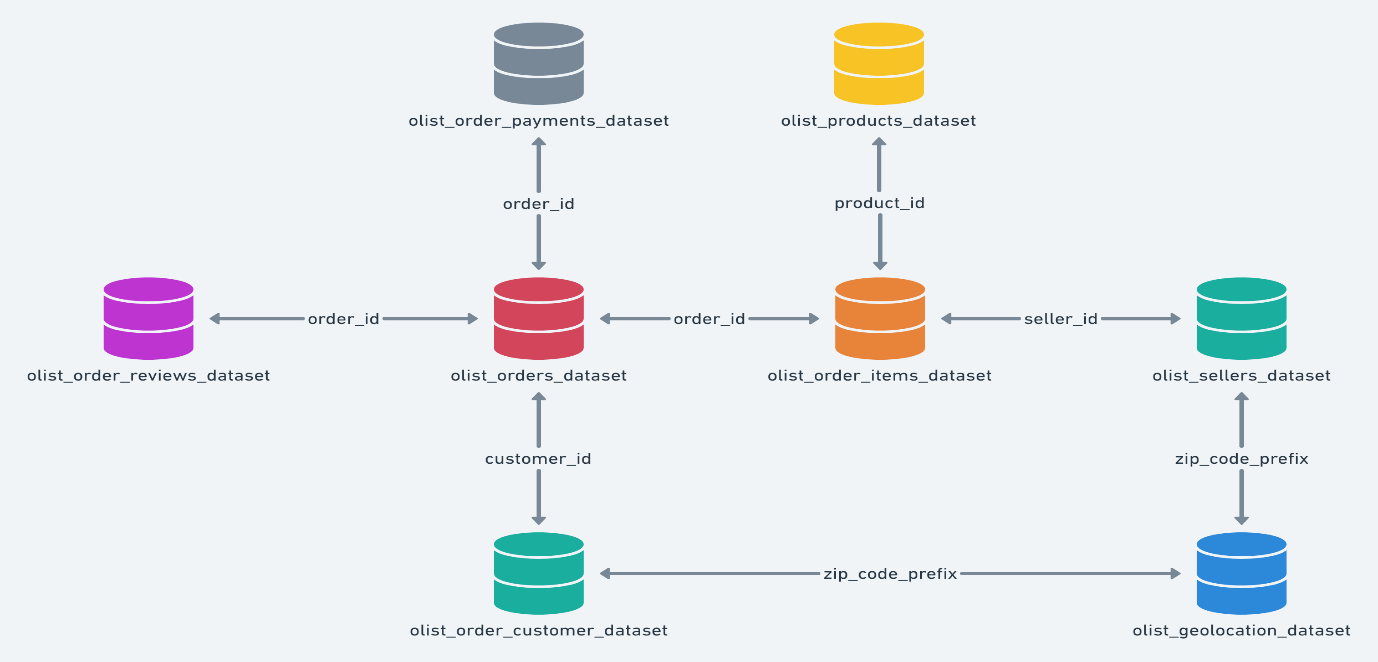
spark.conf.set("fs.azure.account.auth.type.{}.dfs.core.windows.net".format(adls\_account\_name), "OAuth")

spark.conf.set("fs.azure.account.oauth.provider.type.{}.dfs.core.windows.net".format(adls\_account\_name), "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")

spark.conf.set("fs.azure.account.oauth2.client.id.{}.dfs.core.windows.net".format(adls\_account\_name), client\_id)

spark.conf.set("fs.azure.account.oauth2.client.secret.{}.dfs.core.windows.net".format(adls\_account\_name), client\_secret)

spark.conf.set("fs.azure.account.oauth2.client.endpoint.{}.dfs.core.windows.net".format(adls\_account\_name), "https://login.microsoftonline.com/{}/oauth2/token".format(tenant\_id))



1. Create Azure Synapse analytics
   1. Notedown the Name
      1. Provide “Storage Blob Data Contributer” Role to the synapse for storage account
      2. Use Ingest in Home tab to transfer the data
         1. Create LS to ADLS Gen2 🡪 Silver folder
         2. Select Parquet as file format
         3. Create a LS to sink ADLS Gen2 🡪 Gold folder
         4. Click on debug by going to Integrate tab